LUND UNIVERSITY

Accelerated gradient methods and dual decomposition in distributed model predictive control

Giselsson, Pontus; Doan, Dang; Keviczky, Tamas; De Schutter, Bart; Rantzer, Anders

Published in:
Automatica

DOI:
10.1016/j.automatica.2013.01.009

2013

Link to publication

Citation for published version (APA):
Giselsson, P., Doan, D., Keviczky, T., De Schutter, B., & Rantzer, A. (2013). Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, *49*(3), 829-833. https://doi.org/10.1016/j.automatica.2013.01.009

Total number of authors:
5

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Accelerated gradient methods and dual decomposition in distributed model predictive control ⋆

Pontus Giselsson [a], Minh Dang Doan [b], Tamás Keviczky [b], Bart De Schutter [b], Anders Rantzer [a]

[a] *Department of Automatic Control, Lund University, Sweden*

[b] *Delft Center for Systems and Control, Delft University of Technology, The Netherlands*

Abstract

We propose a distributed optimization algorithm for mixed $\mathcal{L}_1/\mathcal{L}_2$-norm optimization based on accelerated gradient methods using dual decomposition. The algorithm achieves convergence rate $O(\frac{1}{k^2})$, where k is the iteration number, which significantly improves the convergence rates of existing duality-based distributed optimization algorithms that achieve $O(\frac{1}{k})$. The performance of the developed algorithm is evaluated on randomly generated optimization problems arising in distributed model predictive control (DMPC). The evaluation shows that, when the problem data is sparse and large-scale, our algorithm can outperform current state-of-the-art optimization software CPLEX and MOSEK.

Key words: Distributed control; Gradient methods; Predictive control.

1 Introduction

Gradient-based optimization methods are known for their simplicity and low complexity within each iteration. A limitation of classical gradient-based methods is the slow rate of convergence. It can be shown [3], [20] that for functions with a Lipschitz-continuous gradient, i.e., smooth functions, classical gradient-based methods converge at a rate of $O(\frac{1}{k})$, where k is the iteration number. In [16] it was shown that a lower bound on the convergence rate for gradient-based methods is $O(\frac{1}{k^2})$. Nesterov showed in his work [17] that an accelerated gradient algorithm can be constructed such that this lower bound on the convergence rate is achieved when minimizing unconstrained smooth functions. This result has been extended and generalized in several publications to handle constrained smooth problems and smooth problems with an additional non-smooth term [18], [19], [2] and [24]. Gradient-based methods are suitable for distributed optimization when they are used in combination with dual decomposition techniques.

Dual decomposition is a well-established concept since around 1960 when Uzawa's algorithm [1] was presented. Similar ideas were exploited in large-scale optimization [6]. Over the next decades, methods for decomposition and coordination of dynamic systems were developed and refined [9], [13], [23] and used in large-scale applications [5]. In [25] a distributed asynchronous method was studied. More recently dual decomposition has been applied in the distributed model predictive control literature in [7], [8], [11] and [15] for problems with a strongly convex quadratic cost and arbitrary linear constraints. The above mentioned methods rely on gradient-based optimization, which suffers from slow convergence properties $O(\frac{1}{k})$. Also the step size parameter in the gradient scheme must be chosen appropriately to get good performance. Such information has not been provided or has been chosen conservatively in these publications.

In this work, we improve on the previously presented distributed optimization methods by using an accelerated gradient method to solve the dual problem instead of a classical gradient method. We also extend the class of problems considered by allowing an additional sparse but non-separable 1-norm penalty. Such 1-norm terms are used as regularization term or as penalty for soft con-

⋆ This paper was not presented at any IFAC meeting. Corresponding author P. Giselsson. Tel. +46 46 222 42 87. Fax +46 46 13 81 18.

Email addresses: pontusg@control.lth.se (Pontus Giselsson), m.d.doan@tudelft.nl (Minh Dang Doan), t.keviczky@tudelft.nl (Tamás Keviczky), b.deschutter@tudelft.nl (Bart De Schutter), rantzer@control.lth.se (Anders Rantzer).

straints [22]. Further, we provide the optimal step size parameter for the algorithm, which is crucial for performance. The convergence rate for the dual function value using the accelerated gradient method is implicitly known from [2], [24]. However, the convergence rate in the dual function value does not indicate the rate at which the primal iterate approaches the primal optimal solution. In this paper we also provide convergence rate results for the primal variables.

Related to our work is the method presented in [14] for systems with a (non-strongly) convex cost. It is based on the smoothing technique presented by Nesterov in [19]. Other relevant work is presented in [12], [21] in which optimization problems arising in model predictive conrol (MPC) are solved in a centralized fashion using accelerated gradient methods. These methods are, however, restricted to handle only box-constraints on the control signals.

To evaluate the proposed distributed algorithm, we solve randomly generated large-scale and sparse optimization problems arising in distributed MPC and compare the execution times to state-of-the-art optimization software for large-scale optimization, in particular CPLEX and MOSEK. We also evaluate the performance loss obtained when suboptimal step lengths are used.

The paper is organized as follows. In Section 2, the problem setup is introduced. The dual problem to be solved is introduced in Section 3 and some properties of the dual function are presented. The distributed solution algorithm for the dual problem is presented in Section 4. In Section 5 a numerical example is provided, followed by conclusions drawn in Section 6.

2 Problem setup

In this paper we present a distributed algorithm for optimization problems with cost functions of the form

$$J(x) = \frac{1}{2}x^T Hx + g^T x + \gamma\|Px - p\|_1. \qquad (1)$$

The full decision vector, $x \in \mathbb{R}^n$, is composed of local decision vectors, $x_i \in \mathbb{R}^{n_i}$, according to $x = [x_1^T, \ldots, x_M^T]^T$. The quadratic cost matrix $H \in \mathbb{R}^{n\times n}$ is assumed separable, i.e., $H = \text{blkdiag}(H_1, \ldots, H_M)$ where $H_i \in \mathbb{R}^{n_i\times n_i}$. Further, H is assumed positive definite with $\underline{\sigma}(H)I \preceq H \preceq \bar{\sigma}(H)I$, where $0 < \underline{\sigma}(H) \leq \bar{\sigma}(H) < \infty$. The linear part $g \in \mathbb{R}^n$ consists of local parts, $g = [g_1^T, \ldots, g_M^T]^T$ where $g_i \in \mathbb{R}^{n_i}$. Further, $P \in \mathbb{R}^{m\times n}$ is composed of $P = [P_1, \ldots, P_m]^T$, where each $P_r = [P_{r1}^T, \ldots, P_{rM}^T]^T \in \mathbb{R}^n$ and $P_{ri} \in \mathbb{R}^{n_i}$. We do not assume that the matrix P should be block-diagonal which means that the cost function J is not separable. However, we assume that the vectors P_r have sparse structure. Sparsity refers to the property that for each $r \in \{1, \ldots, m\}$ there exist some $i \in \{1, \ldots, M\}$ such that $P_{ri} = 0$. We also have $p = [p_1, \ldots, p_m]^T$ and $\gamma > 0$. This gives the following equivalent formulation of (1)

$$J(x) = \sum_{i=1}^{M}\left[\frac{1}{2}x_i^T H_i x_i + g_i^T x_i\right] + \sum_{r=1}^{m}\left|\sum_{i=1}^{M} P_{ri}^T x_i - p_r\right|. \qquad (2)$$

Minimization of (1) is subject to linear equality and inequality constraints

$$A_1 x = B_1, \qquad A_2 x \leq B_2$$

where $A_1 \in \mathbb{R}^{q\times n}$ and $A_2 \in \mathbb{R}^{(s-q)\times n}$ contain $a_l \in \mathbb{R}^n$ as $A_1 = [a_1, \ldots, a_q]^T$ and $A_2 = [a_{q+1}, \ldots, a_s]^T$. Further, each $a_l = [a_{l1}^T, \ldots, a_{lM}^T]^T$ where $a_{li} \in \mathbb{R}^{n_i}$. Further we have $B_1 \in \mathbb{R}^q$ and $B_2 \in \mathbb{R}^{s-q}$ where $B_1 = [b_1, \ldots, b_q]^T$ and $B_2 = [b_{q+1}, \ldots, b_s]^T$. We assume that the matrices A_1 and A_2 are sparse. By introducing the auxiliary variables y and the constraint $Px - p = y$ we get the following optimization problem

$$\begin{aligned} \min_{x,y} \quad & \tfrac{1}{2}x^T Hx + g^T x + \gamma\|y\|_1 \\ \text{s.t.} \quad & A_1 x = B_1 \\ & A_2 x \leq B_2 \\ & Px - p = y. \end{aligned} \qquad (3)$$

The objective of the optimization routine is to solve (3) in a distributed fashion using several computational units, where each computational unit computes the optimal local variables, denoted x_i^*, only. Each computational unit is assigned a number of constraints in (3) for which it is responsible. We denote the set of equality constraints that unit i is responsible for by $\mathcal{L}_i^1$, the set of inequality constraints by $\mathcal{L}_i^2$ and the set of constraints originating from the 1-norm by $\mathcal{R}_i$. This division is obviously not unique but all constraints should be assigned to one computational unit. Further for $l \in \mathcal{L}_i^1$ and $l \in \mathcal{L}_i^2$ we require that $a_{li} \neq 0$ and for $r \in \mathcal{R}_i$ that $P_{ri} \neq 0$. Now we are ready to define two sets of neighbors to computational unit i

$$\begin{aligned} \mathcal{N}_i = \big\{ j \in \{1, \ldots, M\} \mid\ & \exists l \in \mathcal{L}_i^1 \text{ s.t. } a_{lj} \neq 0 \\ & \text{or } \exists l \in \mathcal{L}_i^2 \text{ s.t. } a_{lj} \neq 0 \\ & \text{or } \exists r \in \mathcal{R}_i \text{ s.t. } P_{rj} \neq 0 \big\}, \\ \mathcal{M}_i = \big\{ j \in \{1, \ldots, M\} \mid\ & \exists l \in \mathcal{L}_j^1 \text{ s.t. } a_{li} \neq 0 \\ & \text{or } \exists l \in \mathcal{L}_j^2 \text{ s.t. } a_{li} \neq 0 \\ & \text{or } \exists r \in \mathcal{R}_j \text{ s.t. } P_{ri} \neq 0 \big\}. \end{aligned}$$

Through the introduction of these sets, the constraints that are assigned to unit i can equivalently be written as

$$a_l^T x = b_l \Leftrightarrow \sum_{j\in\mathcal{N}_i} a_{lj}^T x_j = b_l, \qquad l \in \mathcal{L}_i^1 \tag{4}$$

$$a_l^T x \leq b_l \Leftrightarrow \sum_{j\in\mathcal{N}_i} a_{lj}^T x_j \leq b_l, \qquad l \in \mathcal{L}_i^2 \tag{5}$$

and the 1-norm term can equivalently be written as

$$|P_r^T x - p_r| = \Big| \sum_{j\in\mathcal{N}_i} P_{rj}^T x_j - p_r \Big|, \qquad r \in \mathcal{R}_i. \tag{6}$$

In the following section, the dual function to be maximized is introduced. First, we state an assumption that will be useful in the continuation of the paper.

Assumption 1 *We assume that there exists a vector $\bar{x}$ such that $A_1\bar{x} = b_1$ and $A_2\bar{x} < b_2$. Further, we assume that $a_l, l = 1,\dots,q$ and $P_r, r = 1,\dots,m$ are linearly independent.*

Remark 2 *Assumption 1 is known as the Mangasarian-Fromovitz constraint qualification (MFCQ). In [10] it was shown that MFCQ is equivalent to the set of optimal dual variables being bounded. For convex problems, MFCQ is equivalent to Slater's constraint qualification with the additional requirement that the vectors defining the equality constraints should be linearly independent.*

3 Dual problem

In this section we introduce a dual problem to (3) from which the primal solution can be obtained. We show that this dual problem has the properties required to apply accelerated gradient methods.

3.1 Formulation of the dual problem

We introduce Lagrange multipliers, $\lambda \in \mathbb{R}^q, \mu \in \mathbb{R}^{s-q}_{\geq 0}$, $\nu \in \mathbb{R}^m$ for the constraints in (3). Under Assumption 1 it is well known (cf. [4, §5.2.3]) that there is no duality gap and we get the following dual problem

$$\sup_{\lambda,\mu\geq 0,\nu} \inf_{x,y} \Big\{ \frac{1}{2}x^T H x + g^T x + \gamma\|y\|_1 + \lambda^T(A_1 x - B_1) + \\ + \mu^T(A_2 x - B_2) + \nu^T(Px - p - y) \Big\}. \tag{7}$$

After rearranging the terms we get

$$\sup_{\lambda,\mu\geq 0,\nu} \Big\{ \inf_x \Big[(A_1^T\lambda + A_2^T\mu + P^T\nu + g)^T x + \frac{1}{2}x^T H x \Big] \\ - \lambda^T B_1 - \mu^T B_2 - \nu^T p + \inf_y \left[\gamma\|y\|_1 - \nu^T y\right] \Big\}. \tag{8}$$

The infimum over y can be solved explicitly

$$\begin{aligned} \inf_y \left\{\gamma\|y\|_1 - \nu^T y\right\} &= \inf_y \left\{ \sum_i \left(\gamma|[y]_i| - [\nu]_i[y]_i\right) \right\} \\ &= \sum_i \left\{ \inf_{[y]_i} \left(\gamma|[y]_i| - [\nu]_i[y]_i\right) \right\} \\ &= \begin{cases} 0 & \text{if } \|\nu\|_\infty \leq \gamma \\ -\infty & \text{else} \end{cases} \end{aligned}$$

where $[\cdot]_i$ denotes the i-th element in the vector. The infimum over y becomes a box-constraint for the dual variables ν. This is a crucial observation for distribution reasons.

Before we explicitly solve the minimization over x in (8) the following notation is introduced

$$\mathcal{A} = [A_1^T\ A_2^T\ P^T]^T \quad \mathcal{B} = [B_1^T\ B_2^T\ p^T]^T \quad z = [\lambda^T\ \mu^T\ \nu^T]^T$$

where $\mathcal{A} \in \mathbb{R}^{(s+m)\times n}$, $\mathcal{B} \in \mathbb{R}^{s+m}$ and $z \in \mathbb{R}^{s+m}$. We also introduce the set of feasible dual variables

$$Z = \left\{ z \in \mathbb{R}^{s+m} \;\middle|\; \begin{array}{ll} z_l \in \mathbb{R} & l \in \{1,\dots,q\} \\ z_l \geq 0 & l \in \{q+1,\dots,s\} \\ |z_l| \leq \gamma & l \in \{s+1,\dots,s+m\} \end{array} \right\} \tag{9}$$

The minimization over x in (8) can be solved explicitly

$$\inf_x \left[(\mathcal{A}^T z + g)^T x + \frac{1}{2}x^T H x \right] = \\ = -\frac{1}{2}(\mathcal{A}^T z + g)^T H^{-1}(\mathcal{A}^T z + g)$$

and we get the following dual problem

$$\sup_{z\in Z} \Big\{ -\frac{1}{2}(\mathcal{A}^T z + g)^T H^{-1}(\mathcal{A}^T z + g) - \mathcal{B}^T z \Big\}. \tag{10}$$

We introduce the following definition of the negative dual function

$$f(z) := \frac{1}{2}(\mathcal{A}^T z + g)^T H^{-1}(\mathcal{A}^T z + g) + \mathcal{B}^T z.$$

Since f consists of a quadratic term with positive semidefinite hessian and a linear term, f is differentiable and has the following gradient

$$\nabla f(z) = \mathcal{A}H^{-1}(\mathcal{A}^T z + g) + \mathcal{B}. \tag{11}$$

Further, from the min-max theorem we have that the smallest Lipschitz constant, L, to ∇f is $L = \|\mathcal{A}H^{-1}\mathcal{A}^T\|_2$.

4 Distributed optimization algorithm

In this section we show how the accelerated gradient method can be used to distributively solve (3) by minimizing the negative dual function f. The accelerated proximal gradient method for problem (10) is defined by the following iteration as presented in [24, Algorithm 2] and [2, Eq. 4.1-4.3]

$$v^k = z^k + \frac{k-1}{k+2}(z^k - z^{k-1}) \tag{12}$$

$$z^{k+1} = \mathcal{P}_Z\left(v^k - \frac{1}{L}\nabla f(v^k)\right) \tag{13}$$

where $\mathcal{P}_Z$ is the Euclidean projection onto the set Z. Thus, the new iterate, z^{k+1}, is the previous iterate plus a step in the negative gradient direction projected onto the feasible set.

We define the primal iteration $x^k = H^{-1}(-\mathcal{A}^T z^k - g)$. Using this definition, straightforward insertion of v^k into (11) gives

$$\nabla f(v^k) = -\mathcal{A}\left(x^k + \frac{k-1}{k+2}(x^k - x^{k-1})\right) + \mathcal{B}$$

By defining $\bar{x}^k = x^k + \frac{k-1}{k+2}(x^k - x^{k-1})$ and recalling the partition $z = [\lambda^T\ \mu^T\ \nu^T]^T$ and the definition (9) of the set Z, we find that (12)-(13) can be parallelized:

$$x^k = H^{-1}(-\mathcal{A}^T z^k - g) \tag{14}$$

$$\bar{x}^k = x^k + \frac{k-1}{k+2}(x^k - x^{k-1}) \tag{15}$$

$$\lambda_l^{k+1} = \lambda_l^k + \frac{k-1}{k+2}(\lambda_l^k - \lambda_l^{k-1}) + \frac{1}{L}(a_l^T\bar{x}^k - b_l) \tag{16}$$

$$\mu_l^{k+1} = \max\big\{0, \mu_l^k + \frac{k-1}{k+2}(\mu_l^k - \mu_l^{k-1}) + \frac{1}{L}(a_l^T\bar{x}^k - b_l)\big\} \tag{17}$$

$$\nu_r^{k+1} = \min\big\{\gamma, \max\big[-\gamma, \nu_r^k + \frac{k-1}{k+2}(\nu_r^k - \nu_r^{k-1}) + \frac{1}{L}(P_r^T\bar{x}^k - p_r)\big]\big\}. \tag{18}$$

From these iterations it is not obvious that the algorithm is distributed. By partitioning the constraint matrix as

$$\mathcal{A} = [\mathcal{A}_1, \ldots, \mathcal{A}_M]$$

where each $\mathcal{A}_i = [a_{1i}, \ldots, a_{si}, P_{1i}, \ldots, P_{mi}]^T \in \mathbb{R}^{(s+m)\times n_i}$, and noting that H is block-diagonal, the local primal variables are updated according to

$$x_i^k = H_i^{-1}\big(-\mathcal{A}_i^T z^k - g_i\big) \tag{19}$$
$$= -H_i^{-1}\Big[g_i + \sum_{j\in\mathcal{M}_i}\Big[\sum_{l\in\mathcal{L}_j^1} a_{li}\lambda_l^k + \sum_{l\in\mathcal{L}_j^2} a_{li}\mu_l^k + \sum_{r\in\mathcal{R}_j} P_{ri}\nu_r^k\Big]\Big]$$

Thus, each local primal update, x_i^k, can be computed after communication with neighbors $j \in \mathcal{M}_i$. Through (4)-(6) we note that the dual variable iterations can be updated after communication with neighbors $i \in \mathcal{N}_i$. We get the following distributed algorithm.

Algorithm 1 ***Distributed accelerated gradient algorithm***

Initialize $\lambda^0 = \lambda^{-1}, \mu^0 = \mu^{-1}, \nu^0 = \nu^{-1}$ *and* $x^0 = x^{-1}$
In every node, i, the following computations are performed:
For $k \geq 0$

(1) Compute x_i^k *according to* (19) *and set*

$$\bar{x}_i^k = x_i^k + \frac{k-1}{k+2}(x_i^k - x_i^{k-1})$$

(2) Send $\bar{x}_i^k$ *to each* $j \in \mathcal{M}_i$*, receive* $\bar{x}_j^k$ *from each* $j \in \mathcal{N}_i$

(3) Compute λ_l^{k+1} *according to* (16), (4) *for* $l \in \mathcal{L}_i^1$
Compute μ_l^{k+1} *according to* (17), (5) *for* $l \in \mathcal{L}_i^2$
Compute ν_l^{k+1} *according to* (18), (6) *for* $l \in \mathcal{R}_i$

(4) Send $\{\lambda_l^{k+1}\}_{l\in\mathcal{L}_i^1}, \{\mu_l^{k+1}\}_{l\in\mathcal{L}_i^2}$, $\{\nu_r^{k+1}\}_{r\in\mathcal{R}_i}$ *to each* $j \in \mathcal{N}_i$,
receive $\{\lambda_l^{k+1}\}_{l\in\mathcal{L}_j^1}, \{\mu_l^{k+1}\}_{l\in\mathcal{L}_j^2}$ *and* $\{\nu_r^{k+1}\}_{r\in\mathcal{R}_j}$ *from each* $j \in \mathcal{M}_i$

The convergence rates for the dual function f and the primal variables when running Algorithm 1 are stated in the following theorem.

Theorem 3 *Algorithm 1 has the following convergence rate properties:*

(1) Denote an optimizer of the dual problem (10) *as* z^**. The convergence rate is:*

$$f(z^k) - f(z^*) \leq \frac{2L\|z^0 - z^*\|_2^2}{(k+1)^2}, \forall k \geq 1 \tag{20}$$

(2) Denote the unique optimizer of the primal problem as x^**. The rate of convergence for the primal variable is*

$$\|x^k - x^*\|_2^2 \leq \frac{4L\|z^0 - z^*\|_2^2}{\underline{\sigma}(H)(k+1)^2}, \forall k \geq 1 \tag{21}$$

PROOF. Algorithm 1 is a distributed implementation of [24, Algorithm 2] and [2, Eq. 4.1-4.3] applied to minimize f. The convergence rate in argument 1 follows from [24, Proposition 2] and [2, Theorem 4.4].

For argument 2 we get that the necessary and sufficient KKT conditions [4, p. 244] implies $x^* = H^{-1}(-\mathcal{A}^T z^* - g)$ since H is invertible. This leads to

$$\begin{aligned}
\|x^k - x^*\|_2^2 &= \|H^{-1}(\mathcal{A}^T z^k - \mathcal{A}^T z^*)\|_2^2 \\
&\leq \|H^{-1}\| \|\mathcal{A}^T z^k - \mathcal{A}^T z^*\|_{H^{-1}}^2 \\
&= \frac{1}{\underline{\sigma}(H)} (z^k - z^*)^T \mathcal{A} H^{-1} \mathcal{A}^T (z^k - z^*) \\
&= \frac{1}{\underline{\sigma}(H)} \Big((z^k)^T \mathcal{A} H^{-1} \mathcal{A}^T z^k - (z^*)^T \mathcal{A} H^{-1} \mathcal{A}^T z^* - \\
&\qquad - 2(\mathcal{A} H^{-1} \mathcal{A}^T z^*)^T (z^k - z^*) + \\
&\qquad + 2(\mathcal{B} + \mathcal{A} H^{-1} g)^T (z^k - z^k + z^* - z^*) \Big) \\
&= \frac{2}{\underline{\sigma}(H)} \Big(f(z^k) - f(z^*) - \\
&\qquad - (\mathcal{A} H^{-1} (\mathcal{A}^T z^* + g) + \mathcal{B})^T (z^k - z^*) \Big) \\
&= \frac{2}{\underline{\sigma}(H)} \big(f(z^k) - f(z^*) - \nabla f(z^*)^T (z^k - z^*) \big) \\
&\leq \frac{2}{\underline{\sigma}(H)} (f(z^k) - f(z^*)) \leq \frac{4L \|z^0 - z^*\|_2^2}{\underline{\sigma}(H)(k+1)^2}
\end{aligned}$$

where the first inequality comes from the min-max theorem, the equalities are algebra with addition of some zero-terms, the first inequality in the final row is from the first-order optimality condition [20, Theorem 2.2.5], and the final inequality is due to (20). This completes the proof.

5 Numerical example

In this section we evaluate the performance of Algorithm 1. We compare the presented algorithm to state-of-the-art centralized optimization software for large-scale optimization implemented in C, namely CPLEX and MOSEK. We also evaluate the performance loss when using suboptimal step sizes. Our algorithm is implemented on a single processor to be able to compare execution times.

The comparison is made on 100 random optimization problems arising in distributed MPC. A batch of random stable controllable dynamical systems with random structure and random initial conditions are created. The sparsity fraction, i.e., the fraction of non-zero elements in the dynamics matrix and the input matrix, is chosen to be 0.1. We have random inequality constraints that are generated to guarantee a feasible solution and a 1-norm cost where the P-matrix and p-vector are randomly chosen. The quadratic cost matrices are chosen $Q = I$ and $R = I$. Table 1 shows the numerical results obtained running MATLAB on a Linux PC with a 3 GHz Intel Core i7 processor and 4 GB memory. The optimization software used is CPLEX V12.2 and MOSEK 6.0.0.114 that are accessed via the provided MATLAB interfaces.

Table 1
Algorithm comparison with 1-norm cost term and random state and input constraints. Algorithm 1 is implemented in MATLAB while CPLEX and MOSEK are implemented in C.

Alg.	vars./constr.	tol.	# iters		exec (ms)	
			mean	max	mean	max
1 (L)	4320/3231	0.005	69.8	160	253	609
1 (L_1)	4320/3231	0.005	160	420	594	1532
1 (L_F)	4320/3231	0.005	248	640	934	2444
MOSEK	4320/3231	-	-	-	1945	2674
CPLEX	4320/3231	0.005	-	-	1663	2832
1 (L)	2160/1647	0.005	63.8	100	94	200
1 (L_1)	2160/1647	0.005	75.8	180	115	368
1 (L_F)	2160/1647	0.005	121	320	185	488
MOSEK	2160/1647	-	-	-	334	399
CPLEX	2160/1647	0.005	-	-	282	522

The first column specifies the algorithm used where Algorithm 1 is supplemented with the step size used. L is the optimal step size $L = \|\mathcal{A} H^{-1} \mathcal{A}^T\|_2$, $L_F = \|\mathcal{A} H^{-1} \mathcal{A}^T\|_F$ and $L_1 = \sqrt{\|\mathcal{A} H^{-1} \mathcal{A}^T\|_1 \|\mathcal{A} H^{-1} \mathcal{A}^T\|_\infty}$. We compare to the suboptimal step sizes L_1 and L_F since they can be computed in distributed fashion. The step sizes satisfy $L \leq L_1$ and $L \leq L_F$. The second column specifies the number of variables and constraints in the optimization problems. In the third column we have information about the duality gap tolerance that is used as stopping condition in the algorithms (if possible to set). The two final columns present the results in terms of number of iterations and execution time. The difference between the upper and lower halves of the table is the size of the problems that are solved.

Table 1 reveals that Algorithm 1 performs better than CPLEX and MOSEK on these large-scale sparse problems despite the fact that CPLEX and MOSEK are implemented in C and Algorithm 1 is implemented in MATLAB. We also conclude that the choice of step size in Algorithm 1 is important for performance reasons.

6 Conclusions

We have presented a distributed optimization algorithm for strongly convex optimization problems with sparse problem data. The algorithm is based on an accelerated gradient method that is applied to the dual problem. The

algorithm was applied to large-scale sparse optimization problems originating from a distributed model predictive control formulation. Our algorithm performed better than state-of-the-art optimization software for large-scale sparse optimization, namely CPLEX and MOSEK, on these problems.

Acknowledgements

The second author would like to thank Quoc Tran Dinh for helpful discussions on the topic of this paper.

The second, third and fourth authors were supported by the European Union Seventh Framework STREP project "Hierarchical and distributed model predictive control (HD-MPC)", contract number INFSO-ICT-223854, and the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 257462 HYCON2 Network of Excellence.

The first and last authors were supported by the Swedish Research Council through the Linnaeus center LCCC.

References

[1] K.J. Arrow, L. Hurwicz, and H. Uzawa. Studies in Linear and Nonlinear Programming. Stanford University Press, 1958.

[2] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sciences, 2(1):183–202, October 2009.

[3] Dimitri P. Bertsekas. Nonlinear Programming. Athena Scientific, Belmont, MA, 2nd edition, 1999.

[4] Stephen Boyd and Lieven Vandenberghe. Convex Optimization. Cambridge University Press, New York, NY, 2004.

[5] Pierre Carpentier and Guy Cohen. Applied mathematics in water supply network management. Automatica, 29(5):1215–1250, 1993.

[6] G.B. Danzig and P. Wolfe. The decomposition algorithm for linear programming. Econometrica, 4:767–778, 1961.

[7] M. D. Doan, T. Keviczky, and B. De Schutter. An iterative scheme for distributed model predictive control using Fenchel's duality. Journal of Process Control, 21(5):746–755, June 2011. Special Issue on Hierarchical and Distributed Model Predictive Control.

[8] M. D. Doan, T. Keviczky, I. Necoara, M. Diehl, and B. De Schutter. A distributed version of Han's method for DMPC using local communications only. Control Engineering and Applied Informatics, 11(3):6–15, 2009.

[9] W. Findeisen. Control and Coordination in Hierarchical Systems. International series on applied systems analysis. Wiley, 1980.

[10] Jacques Gauvin. A necessary and sufficient regularity condition to have bounded multipliers in nonconvex programming. Mathematical Programming, 12:136–138, 1977.

[11] P. Giselsson and A. Rantzer. Distributed model predictive control with suboptimality and stability guarantees. In Proceedings of the 49th Conference on Decision and Control, pages 7272–7277, Atlanta, GA, December 2010.

[12] M. Kögel and R. Findeisen. A fast gradient method for embedded linear predictive control. In Proceedings of the 18th IFAC World Congress, pages 1362–1367, Milan, Italy, 2011.

[13] M. D. Mesarovic, D. Macko, and Y. Takahara. Theory of Hierarchical Multilevel Systems. Academic Press, New York, 1970.

[14] I. Necoara and J. Suykens. Application of a smoothing technique to decomposition in convex optimization. IEEE Transactions on Automatic Control, 53(11):2674–2679, December 2008.

[15] R.R. Negenborn, B. De Schutter, and J. Hellendoorn. Multi-agent model predictive control for transportation networks: Serial versus parallel schemes. Engineering Applications of Artificial Intelligence, 21(3):353–366, April 2008.

[16] A Nemirovsky and D Yudin. Informational Complexity and Efficient Methods for Solution of Convex Extremal Problems. Wiley, NewYork, NY, 1983.

[17] Y Nesterov. A method of solving a convex programming problem with convergence rate O $(1/k^2)$. Soviet Mathematics Doklady, 27(2):372–376, 1983.

[18] Y Nesterov. On an approach to the construction of optimal methods of minimization of smooth convex functions. Ékonom. i. Mat. Metody, 24:509–517, 1988.

[19] Yu Nesterov. Smooth minimization of non-smooth functions. Math. Program., 103(1):127–152, May 2005.

[20] Yurii Nesterov. Introductory Lectures on Convex Optimization: A Basic Course (Applied Optimization). Springer Netherlands, 1st edition, 2004.

[21] S. Richter, C.N. Jones, and M. Morari. Real-time input-constrained MPC using fast gradient methods. In Proceedings of the 48th Conference on Decision and Control, pages 7387–7393, Shanghai, China, December 2009.

[22] C. Savorgnan, C. Romani, A. Kozma, and M. Diehl. Multiple shooting for distributed systems with applications in hydro electricity production. Journal of Process Control, 21(5):738–745, 2011.

[23] Madan G. Singh and Andre Titli. Systems: Decomposition, Optimisation, and Control. Pergamon, 1978.

[24] Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. Submitted to SIAM J. Optim. Available: `http://www.math.washington.edu/~tseng/papers/apgm.pdf`, May 2008.

[25] J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. IEEE Transactions on Automatic Control, 31(9):803–812, September 1986.